

intro to ML Final Project

109550043 王健哲

Github link: https://github.com/a15923647/NYCU_2022_CSCS20024_final_project

Models link: https://drive.google.com/drive/folders/15ZMkigpwADFoD_pkejQzDQlvoChkUiSt?usp=sharing

Score: **0.59472**



109550043.csv

Complete (after deadline) · now · should be 0.59472

0.59472

0.57582



Introduction:

在這個比賽中，訓練資料包括 label 有 26 維，經過檢視後發現測試資料中的 "product_code" 訓練資料完全沒有，"attribute_1" 中的選項種類也有缺少，因此訓練參考的屬性扣除 "id"、"attribute_1"、"product_code" 和 "failure" 只有 22 維，屬於較小的資料集，於是嘗試了多種線性架構 (imbalance-ensemble 的所有 classifier、LinearDiscriminantAnalysis)，結果都不是很好，猜測是特徵與結果的線性關聯度較低 (相關度如下圖)，於是決定使用 pytorch 寫一個 DNN。

```
0 th 相關係數: -0.007545109781745928
1 th 相關係數: 0.12516236407261164
2 th 相關係數: 0.00633697502185357
3 th 相關係數: -0.019222134331016875
4 th 相關係數: 0.009645933201494662
5 th 相關係數: -0.01080998860865806
6 th 相關係數: 0.01580756575056424
7 th 相關係數: 0.015520258527871462
8 th 相關係數: -0.012951301860425622
9 th 相關係數: -0.009415009204640338
10 th 相關係數: 0.003643078690638206
11 th 相關係數: 0.007832716916438961
12 th 相關係數: 0.006526392107410167
13 th 相關係數: -0.010302582275531083
14 th 相關係數: -0.0007838760169723551
15 th 相關係數: -0.0010693227382288946
16 th 相關係數: -0.0033991513254432496
17 th 相關係數: 0.0009905066777939249
18 th 相關係數: -0.0029650524796834415
19 th 相關係數: -0.0021329773393567175
20 th 相關係數: 0.0047660206610333645
21 th 相關係數: 0.012883341290169482
22 th 相關係數: 1.0
```

Methodology:

由於可使用的資料維度較上次少，DNN 的特徵輸出維度都沒設定超過 64 維，總共嘗試了兩個模型架構，架構如下。每個架構的訓練流程包含資料前處理、訓練、超參數搜尋。

模型一	模型二
BatchNorm1d(in_features)	BatchNorm1d(in_features)
Linear(in_features, 64)	Linear(in_features, 24)
ReLU()	ReLU()
Linear(64, 32)	Linear(24, 12)
ReLU()	ReLU()
Linear(32, 1)	Linear(12, 1)
Sigmoid()	Sigmoid()

資料前處理的部分，首先處理 class imbalance 的問題，參考[1]ROS(Random Over Sampling)比 RUS(Random Under Sampling)效果較好的結論，將整個訓練資料集先透過 ROS 的方式將每種 class 的資料筆數提升到 1:1；另外這步也有嘗試使用另一種方式[2]SMOTE，利用插值生成新的樣本，但做下來的結果並不理想，最好的 score 不到 0.54，推測應該是資料不太具備連續性。處理完 imbalance 問題，接著和上個作業一樣依 7:3 分為訓練集和驗證集。採用 sklearn.preprocessing.StandardScaler 對數據做標準化，使數據的平均值為 0、標準差為 1，以利 gradient descent。另外在 data 有缺失的部分，以 0 取代，參考[3]並新建一欄<attribute name>_isnan 中紀錄該值是否原本為 nan，這個做法是為了避免網路真的把這個不存在的值和其他有效值做關聯，同時也不會有資訊消失。

訓練過程分為兩個階段。第一階段使用訓練集訓練 model 大概 100 epochs，保存採用驗證集算出來 loss 最小的 model。第二階段使用整個訓練資料集下去訓練 15 epochs，由於不知 model 是否出現 overfitting，這裡每個 epoch 的 model 我都有存下來。訓練結束後串接 kaggle API 將剛剛第二階段的 model 上傳察看結果。

超參數搜尋是透過 shell script 修改 config.py 裡的 learning rate 和 batch size 重複執行上述訓練完成的。具體來說是用 learning rate = ["5e-4" "5e-3" "5e-2" "5e-1"] 和 batch size = [64 128 256 512]先做初步搜尋，取得 batch size 和 learning rate 大致的 scale。再把該 scale 的值依區域畫分成十等分搜尋過一遍。最終發現模型一在 learning rate 在[0.04, 0.06]間、batch size 在[256, 350]間會有比較好的結果；模型二在 learning rate 在[0.03, 0.04]間、batch size 在[128, 256]間會有比較好的結果。

模型一

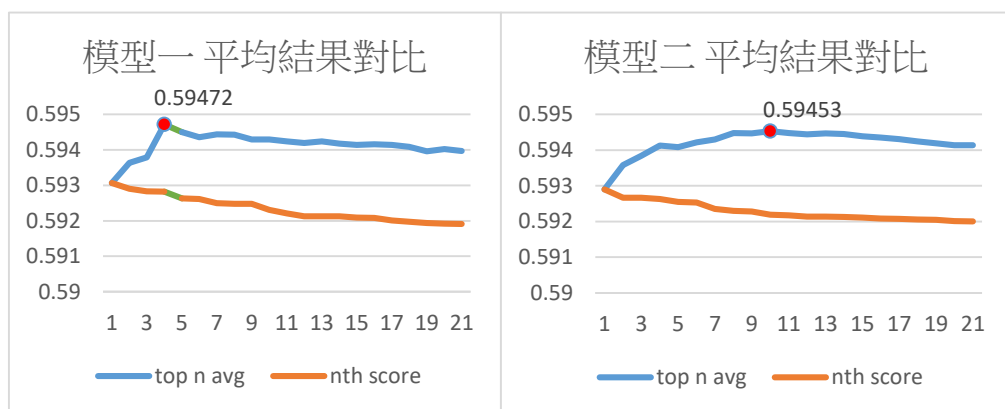
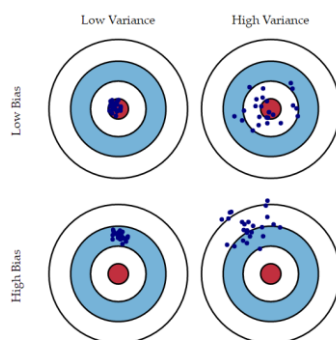
lr / batch size	64	128	256	512
5.00E-01	0.50027	0.54435	0.58365	0.58747
5.00E-02	0.58929	0.59017	0.5903	0.59028
5.00E-03	0.58756	0.58418	0.57931	0.57969

5.00E-04	0.58153	0.57892	0.58059	0.57719
----------	---------	---------	---------	---------

模型二

lr / batch size	64	128	256	512
5.00E-01	0.5	0.50039	0.57918	0.58956
5.00E-02	0.59026	0.59118	0.59058	0.59022
5.00E-03	0.58734	0.58359	0.57521	0.58414
5.00E-04	0.58174	0.57926	0.57563	0.57721

測試方法是將 **score** 較高的模型(test dataset 表現好的)預測結果取平均，以此降低單一模型的 **bias error**，想法如下圖右上的 case。



實測結果如上圖，隨著包含的模型量越多，平均 **bias error** 下降量比平均 **variance error** 上升量大，因此平均後的分數確實有提升；然而隨著包含的模型越不精確，平均 **variance error** 上升量超過平均 **bias error** 下降量，因此分數下降，尤其是模型一中第五高分比第四高分低得多，結果就是平均後的分數陡然下降。

測試結果

● 模型一

■ 單一模型: score: 0.59307, learning rage: 0.06, batch size: 350

	submission_pred_v1.0_improved_14_train_loss_0.68481.model.csv	0.59307	0.57925	<input type="checkbox"/>
	Complete (after deadline) · 8h ago · /mnt/nas/data/study/intro_to_ML/HW/final_project6/DNN_model_desc_lr_6e-2_bs_350_tim...			

■ 平均結果(最佳): score: 0.59472


	submission_p6_top4_avg_at_1200.csv	0.59472	0.57582	<input type="checkbox"/>
	Complete (after deadline) · 28m ago			

● 模型二

■ 單一模型: score: 0.5929, learning rate: 0.035, batch size: 150

	submission.csv	0.5929	0.57016	<input type="checkbox"/>
	Complete (after deadline) · 2d ago · /mnt/nas/data/study/intro_to_ML/HW/final_project3_2/DNN_model_detail_lr_0.035_bs_150...			

■ 平均結果: score: 0.59453

	submission_avg.csv	0.59453	0.57577	<input type="checkbox"/>
	Complete (after deadline) · 20h ago · 3_2 top10: 0.5929, 0.59267, 0.59267, 0.59263, 0.59255, 0.59253, 0.59235, 0.5923, 0.592...			

Summary:

這次 project 中，採用的不是像上個作業利用電腦生成出來的資料；而是真實世界的資料作訓練。於是就會碰到要解決資料缺失和 imbalance class 的問題，另外還要剔除無關的 column，資料前處理就變得尤為重要，資料缺失方面我使用[3]不同的資料表達方式，imbalance class 方面則使用[1]Random Over Sampling。

在模型架構的採用方面，一開始使用 imbalanced ensembling 內的 tree-based classifier 和 LinearDiscriminantAnalysis，由於特徵與結果的線性相關度低，結果不是很好。於是採用 DNN 作為模型架構。訓練過程為避免 overfitting，先採用 30%的驗證資料集做初步訓練，而後再用整個資料及訓練數個 epoch。接著調整 learning rate 和 batch size 來找出合適的參數區間。

產生結果的部分選 test dataset 中表現較佳的 model 做平均來消除單一 model 的 bias error。

比較兩種模型的結果發現兩種模型的表現差異不大，這說明了模型並非參數量越多越好，反而是訓練資料的品質在這次 project 中對分數的影響比較劇烈。

Reference

[1] Deep Learning and Data Sampling with Imbalanced Big Data

https://www.researchgate.net/publication/335937033_Deep_Learning_and_Data_Sampling_with_Imbalanced_Big_Data

[2] SMOTE: synthetic minority over-sampling technique

<https://dl.acm.org/doi/10.5555/1622407.1622416>

[3] Missing values have predictive value

<https://www.kaggle.com/competitions/tabular-playground-series-aug->

[2022/discussion/342319](#)